UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/737,528	12/13/2000	Paul F. Austin	5150-50900	9832

7590 01/13/2004

Jeffrey C. Hood
Conley, Rose & Tayon, P.C.
P.O. Box 398
Austin, TX 78767-0398

EXAMINER	
BASOM, BLAINE T	
ART UNIT	**PAPER NUMBER**
2173	6

DATE MAILED: 01/13/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/737,528	AUSTIN, PAUL F.	
	Examiner	Art Unit	
	Blaine Basom	2173	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *08 October 2003*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-16,19-24 and 27-34* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 4-16, 19-24, and 27-34* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

DETAILED ACTION

Response to Arguments

The Examiner acknowledges the Applicant's amendments to various claims of the present application, the amended claims expressing, in part, a graphical program which includes a block diagram comprising a plurality of connected nodes, wherein the connected nodes visually represent functionality of the graphical program, and wherein this block diagram is automatically configured based on user input specifying a data source or data target. The Applicant subsequently argues that Risberg (U.S. Patent No. 5,339,392) does not teach or suggest a graphical program, nonetheless a method for configuring such a graphical program, as was suggested by the Examiner in the previous Office Action. The Examiner respectfully disagrees with this argument. As the program presented by Risberg comprises graphics, such as various graphical user interface elements (for example, see figure 1 and its associated description), it is understood that, given the broadest most reasonable definition of a "graphical program," Risberg in fact presents such a program. As also shown in the previous Office Action, and again below, Risberg teaches a technique for configuring these graphical user interface elements to receive and display data. Risberg is consequently considered to present a method for configuring a graphical program. Further regarding the amended claims of the present application, the Applicant submits that Risberg does not disclose that this program comprises a block diagram having a plurality of connected nodes, wherein the connected nodes visually represent functionality of the graphical program, and wherein this block diagram is automatically configured. The Examiner agrees with the Applicant in that Risberg does not present such a block diagram. Nevertheless, Kodosky (U.S. Patent No. 5,291,587) presents such a block

diagram and suggests including this block diagram in a graphical program similar to that described by Risberg, as was shown in the previous Office Action and is again shown below. The Examiner thus submits that the combination of Risberg and Kodosky presents a graphical program which includes a block diagram comprising a plurality of connected nodes, wherein the connected nodes visually represent functionality of the graphical program, and wherein this block diagram is automatically configured based on user input specifying a data source or data target.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1, 4-11, 14-15, 24, and 27-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,339,392, which is attributed to Risberg et al. (and hereafter referred to as "Risberg"), and also over U.S. Patent No. 5,291,587, which is attributed to Kodosky et al. (and hereafter referred to as "Kodosky"). In general, Risberg discloses an application to be used for monitoring and managing complex systems having a plurality of frequently varying data values. More specifically, and regarding the claimed invention, this application allows users to create custom graphical user interfaces in which these data values are displayed, and in which changes in these data values are immediately reflected on the display

(see column 1, lines 31-41). These data values, which are received from one or more sources over a network, are particularly displayed via one or more GUI elements. It is thus understood that Risberg teaches a method for configuring a GUI element to publish or subscribe to a data target or data source.

As per claims 1, 24, and 31, the application disclosed by Risberg is used to construct GUIs, referred to as "active documents," which are for viewing financial information such as stock prices. Risberg however notes that the application also applies to *any* system which generates real time data that must be monitored (see column 2, lines 52-55). In any event, Risberg discloses that the data to be monitored is displayed by a plurality of GUI elements, specifically "quotes," "dynamic graphs," "tickers," or "page fragments" (see column 28, lines 30-67). For example, a page fragment displays a section of data obtained from a financial data source. The data displayed via a fragment element is updated in real time (see column 28, lines 62-67). Quotes, dynamic graphs, and tickers similarly display data from one or more financial sources, except in a different format. To create a page fragment on the active document, a user uses a "Page Fragment tool" and drags, with a mouse, a region on the active document where the page fragment is to be positioned. In response, the page fragment is displayed but contains no information (see column 11, line 65 – column 12, line 4). For the page fragment to display information, the user enters a data source, i.e. "service," into a specific dialog box provided to the user, wherein the data service provides financial data which is displayed in the page fragment (see column 12, lines 5-28). As this data displayed by the page fragment is updated in real time (see column 28, lines 62-67), it is understood that the page fragment is thus configured to receive and display data from the specified data source. It is interpreted that quotes, dynamic graphs,

and tickers are created and configured by similar means. Lastly, Risberg notes that the data displayed by a quote, dynamic graph, ticker, or page fragment can be published on a network so that it may be used as a bulletin board or by other users linked to the network (see column 3, line 66 – column 4, line 4). Thus regarding claim 1, Risberg teaches displaying a GUI element, such as a page fragment, on a display; and, in response to receiving user input specifying at least one of a data source or data target with which to associate the GUI element, automatically configuring the GUI element to perform at least one of: receiving and displaying data from the specified data source; and/or publishing data associated with the GUI element to the specified data target. Specifically referring to claims 24 and 31, the above-described method taught by Risberg is implemented as a software program (for example, see column 26, lines 53-68). Consequently, it is understood that it is executed by a computer system, the computer system having: a display device to present the active document to the user; a processor to execute the software program; and a memory coupled to the processor to store the program, as is known in the art. Such a computer system implementing the above-described method of Risberg is considered a system, like that recited in claim 24, which is for configuring a GUI element to publish or subscribe to a data target or source. Similarly, the computer memory storing this software program of Risberg is considered a memory medium, like that recited in claim 31, which is for configuring a GUI element to publish or subscribe to a data target or source. As shown above, Risberg presents an active document containing one or more GUI elements, which are automatically configured to receive and display data from user-specified data sources. Since an active document contains GUI elements, it is considered a graphical program. Moreover, Risberg discloses that a user may create one or more scripts which define the functionality of the

active document, whereby the scripts access this data provided by the GUI elements and thus function according to this data. For example, Risberg discloses that a script may be created, which is executed if the data received by a quote element is in a pre-defined "normal" state. On the other hand, if the data received by a quote element is in a pre-defined "alert" state, an alternative script is executed (see column 10, lines 4-27). These scripts may perform one or more operations, including selecting user-specified objects and editing these objects (see column 16, lines 62-68). However, and with respect to the claimed invention, Risberg does not explicitly disclose that the active document includes a block diagram comprising a plurality of connected nodes, wherein as recited in each of claims 1, 24, and 31, the connected nodes visually represent functionality of the active document. Consequently, Risberg also does not explicitly disclose that this block diagram is automatically configured based on user input specifying a data source or data target as is further recited in each of claims 1, 24, and 31.

Like Risberg, Kodosky presents an application that is used to construct graphical user interfaces, referred to as a "front panels," which provide for monitoring and managing complex systems having a plurality of frequently varying data values. Such complex systems are namely data collecting instruments (see column 2, lines 6-18 and lines 44-52). Regarding the claimed invention, Kodosky teaches that a "graphical editor" may be used to construct a graphical diagram to specify the functionality of the interface (see column 7, lines 36-40). Figure 22 shows an example of such a functional diagram. As shown in figure 22, the functional diagram is essentially a block diagram comprising a plurality of connected nodes, each node representing a particular function or element of the front panel. As further taught by Kodosky, when a GUI

element is placed in a front panel, a node representative of the GUI element is automatically placed in the functional diagram (see column 14, lines 59-64).

It would therefore have been obvious to one of ordinary skill in the art, having the teachings of Risberg and Kodosky before him at the time the invention was made, to modify the application taught by Risberg such that, like the application taught by Kodosky, a user may create a block diagram to define the functionality of the active document created by the application, instead of creating scripts. It would have been advantageous to one of ordinary skill to utilize such combination because an easier means for defining the functionality of the active document would have been obtained, as is taught by Kodosky (see column 2, lines 53-65, and column 3, lines 30-35). Consequently, with this combination of Risberg and Kodosky, an active document would comprise a block diagram, the block diagram including a plurality of connected nodes, wherein the connected nodes visually represent functionality of the active document. And since each GUI element in the active document is represented by a node, automatically configuring a GUI element to receive data from a user-specified data source, as Risberg teaches, is equivalent to configuring the node representing the GUI element to receive data from the user-specified data source. Thus the block diagram is automatically configured based on user input specifying a data source or data target.

Regarding claims 4 and 27, the combination of Risberg and Kodosky teaches a method whereby an application is used to construct a GUI, namely an active document, as is shown above. More particularly, Risberg teaches that this application is used to display and configure a GUI element to receive and display data from a specific data source. Since the GUI element is automatically configured and displayed by an application, i.e. program, the GUI element is

considered to be programmatically displayed and configured. Thus the combination further teaches automatically configuring and displaying a GUI element by programmatically configuring the graphical program. In addition, since the GUI element is automatically configured and displayed by the program, it is also understood that the graphical program is configured without user input.

As per claims 5, 28, and 32, with the above-described combination of Risberg and Kodosky, an active document comprises a block diagram, wherein the block diagram includes a plurality of connected nodes, and wherein the connected nodes visually represent functionality of the active document. Particularly, it is understood that each GUI element in the active document is represented by a node in the block diagram (for example, see column 14, lines 52-64 of Kodosky). And since each GUI element in the active document is represented by a node, automatically configuring a GUI element to receive data from a user-specified data source, as Risberg teaches, necessitates configuring the node representing the GUI element to receive data from the user-specified data source. Consequently, the combination of Risberg and Kodosky teaches automatically configuring one or more nodes to perform at least one of the following during program execution: receiving data from a specified data source, or publishing data to a specified data target.

In regard to claims 6 and 33, with the above-described combination of Risberg and Kodosky, an active document comprises a graphical diagram, wherein as shown above, the graphical diagram includes a plurality of connected nodes, the connected nodes visually representing the functionality of the active document. Kodosky further teaches that these nodes may be selected in order to move them around the graphical diagram (see column 23, lines 5-23).

It is thus understood that a node corresponding to a GUI element in the active document may be selected. And since each GUI element in the active document is represented by a node, automatically configuring a GUI element to receive data from a user-specified data source, as Risberg teaches, is equivalent to configuring the node representing the GUI element to receive data from the user-specified data source. Consequently the above-described combination of Risberg and Kodosky further teaches receiving user input selecting a first node of the plurality of nodes; and automatically configuring the first node to programmatically receive data from a specified data source.

Referring to claim 7, the GUI elements disclosed by Risberg are implemented by objects, speaking with respect to the well-known object-oriented programming paradigm (see column 20, lines 53-64). Such objects are implemented as a data structure in memory (see column 19, lines 20-23). To configure such GUI element to receive and display data from a specific data source, a user enters the specific data source into a dialog box associated with the object (for example, see column 11, line 65 – column 12, line 27). In response, it is interpreted that the object implementing the GUI element is similarly updated to reflect the data source input by the user (see column 24, lines 41-64). Consequently, the combination of Risberg and Kodosky further teaches automatically creating and storing a data structure, specifically an object, wherein the data structure comprises source information, the source information useable during execution of the active document to receive data from the specified data source.

As for claims 8, 29, and 34 with the above-described combination of Risberg and Kodosky, an active document comprises a graphical diagram, i.e. block diagram. As shown above, this graphical diagram includes a plurality of connected nodes, the connected nodes

visually representing the functionality of the active document. As is further taught by Kodosky, when a GUI element is placed in an active document, i.e. front panel, a node representative of the GUI element is automatically placed in the graphical diagram (see column 14, lines 59-64). Thus creating and configuring a GUI element to receive and display data from a specific data source, as Risberg teaches, implies automatically including a node in a block diagram of the active document, the node representing the GUI element. And since each GUI element in the active document is represented by a node, automatically configuring a GUI element to receive data from a user-specified data source is equivalent to configuring the node representing the GUI element to receive data from the user-specified data source. Thus the above-described combination of Risberg and Kodosky teach automatically including one or more nodes in a block diagram of an active document, wherein during execution of the graphical program, the one or more nodes are operable to receive data from a specific data source.

Concerning claim 9, with the above-described combination of Risberg and Kodosky, each GUI element in an active document is represented by a node in a block diagram. As is shown above in the rejection for claim 8, such nodes may be configured to receive data from a user-specified data source. Consequently, these nodes are considered equivalent to the "DataSocket" nodes recited in claim 9.

With respect to claim 10, figure 22 of Kodosky shows an example of a block diagram representing a graphical user interface. As shown in figure 22, the block diagram comprises a plurality of connected nodes, each node representing a particular function or element of the graphical user interface. Thus with the above-described combination of Risberg and Kodosky, it is understood that the block diagram may comprise at least two nodes. Each GUI element in an

active document is represented by such a node in a block diagram, whereby as shown above, such nodes may be configured to receive data from a user-specified data source. As shown in figure 22 of Kodosky, the nodes are connected. Since a node representative of a GUI element is operable to receive data from a specified data source, it is understood that any set of two nodes comprising a node representative of a GUI element would likewise be operable to receive data from the specified data source. And finally, since the block diagram visually indicates the functionality of an active document, as is stated above, it is understood that a node representative of a GUI element configured to receive data from a specified data source would visually indicate receiving data from the specified data source. Thus since one such node visually indicates the reception of data from a specified data source, it is understood that any set of two nodes comprising such a node would likewise indicate the reception of data from the specified data source. The above-described combination of Risberg and Kodosky therefore teaches: automatically including at least two nodes in a block diagram; automatically connecting the at least two nodes such that the two connected nodes are operable to receive data from a specified data source; and wherein the at least two connected nodes visually indicate receiving data from the specified data source.

Referring to claim 11, the application disclosed by Risberg is used to construct GUIs, or more specifically, active documents. An active document is considered a computer program, as it is executed on a computer to access data over a network and display it on the computer. Thus the page fragments, quotes, dynamic graphs, and tickers described by Risberg are GUI elements associated with a first computer program, specifically an active document. As shown above, these GUI elements are operable to receive and display data from a specific data source during

execution of the active document. Thus the combination of Risberg and Kodosky teaches executing a graphical program after configuring the GUI elements of the program, the graphical program receiving data from a user-specified data source.

With respect to claim 14, Risberg discloses that a data source may be specified by mouse actions equivalent to a drag and drop technique (see column 11, lines 46-64).

With respect to claim 15, Risberg further discloses that a data source may be specified with a dialog box (see column 11, line 65 – column 12, line 27).

In regard to claim 30, the above-described combination of Risberg and Kodosky teach automatically including one or more nodes in a block diagram of an active document, wherein as shown above in the rejection for claim 29, the one or more nodes are operable to receive data from a specific data source. Since the block diagram visually indicates the functionality of an active document, as is stated above, it is understood that a node representative of a GUI element configured to receive data from a specified data source would visually indicate receiving data from the specified data source.

Claims 12 and 13 are rejected under 35 U.S.C. 103(a) as being obvious over the combination of Risberg and Kodosky, which is described above, and also over U.S. Patent No. 5,959,621, which is attributed to Nawaz et al. (and hereafter referred to as "Nawaz"). As shown above, the combination of Risberg and Kodosky teaches a method like that of claim 1, wherein a user enters a data source into a dialog box. In response, a GUI element is automatically configured to receive and display data from this data source, as is shown above. Neither Risberg

nor Kodosky, however, explicitly disclose that this input specifying a data source is a URL, as is expressed in claim 12.

Like Risberg and Kodosky, Nawaz presents a method for configuring a GUI element, specifically a ticker, to receive and display data from a specific data source. With further similarity to the teachings of Risberg, this data may comprise financial data, namely stock prices (see column 3, lines 27-29). Regarding the claimed invention, Nawaz teaches that the data is received and displayed in the ticker from sources specified by URLs (see column 12, lines 23-40).

It would have therefore been obvious to one of ordinary skill in the art, having the teachings of Risberg, Kodosky, and Nawaz before him at the time the invention was made, to modify the method taught by Risberg and Kodosky such that the data sources are specified by URLs, as is taught by Nawaz. One would have been motivated to create such a combination because, as is demonstrated by Nawaz, URLs provide well-known and commonly used identification means for identifying data on a network.

With respect to claim 13, Risberg discloses that a data source may be specified by selecting the particular source via a drag and drop technique (see column 11, lines 46-64). It is understood that some identification of this particular source is stored in memory (see column 20, lines 53- 64, and column 19, lines 20-23). Consequently, with the above-described combination of Risberg, Kodosky, and Nawaz, it is interpreted that the URL of this data source would be automatically generated in response to such selection means, and then stored in memory.

Claims 16, and 19-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over the combination of Risberg and Kodosky, which is described above, and also over Microsoft Office 97, as is described by Lonnie E. Moseley and David M. Boodey in the book entitled *Mastering Microsoft Office 97, Professional Edition* (which is hereafter referred to as "Office 97"). As shown above in the rejection regarding claim 1, the combination of Risberg and Kodosky teaches a method involving: displaying a GUI element, such as a page fragment, as a component of an active document; receiving user input, specifically through a dialog box, wherein this user input specifies a data source with which to associate the GUI element; and, in response to receiving this input, automatically configuring the GUI element to receive and display data from the specified data source. As is further described above, this GUI element is represented as a node in a block diagram, wherein the block diagram comprises a plurality of connected nodes, and wherein the connected nodes visually represent functionality of the active document. Consequently it is understood that the user input specifying the data source is received by the block diagram in order for the block diagram to appropriately represent functionality of the active document. Thus the above-described combination of Risberg and Kodosky teaches: receiving user input specifying a data source, wherein the user input is received to a block diagram of the graphical program; automatically configuring a GUI element to receive and indicate data from the specified data source during execution of the program; and wherein automatically displaying and automatically configuring are performed based on the user input specifying the data source. However, and with respect to the claimed invention, Risberg teaches displaying the GUI element prior to receiving user input specifying at least one of a data source or target. Consequently, the combination of Risberg and Kodosky does not teach automatically

displaying a GUI element in response to the user input, as is expressed in claim 16 and more specifically in claims 19-23.

Similar to the application taught by Risberg, which is used to construct active documents, i.e. GUIs, Microsoft Word is an application used to construct documents. More specifically, these Word documents created by Microsoft word may comprise elements, which like the page fragments taught by Risberg and described above, display information obtained from one of various external sources such as spreadsheets or other documents (see pages 44 and 45 of Office 97). Also like the page fragment of Risberg, this information displayed by the element in the Microsoft Word document may be linked to the data source such that live data from the source is used. In other words, if the data changes at the source, the data displayed by the element of the Microsoft Word document similarly changes (see pages 44 and 45 of Office 97). With respect to the claimed invention, a user specifies a data source by selecting the data source with a mouse cursor and then selecting a "paste" option in the "edit" menu of Microsoft Word. In response, the selected data source information is displayed in an appropriate GUI element in the Microsoft Word document. For example, when pasting a spreadsheet source, the data appears in the Microsoft Word document via a table-like GUI element (see figure 3.9 on page 46 of Office 97). Moreover, and like the teachings of Risberg, it is understood that the user may also specify a particular data source by selecting the source in an "Insert File" dialog box provided to the user, as apposed to copying and pasting the source. The selected source data is displayed in the Word document in response (see pages 380 and 381 of Office 97). In any event, Office 97 teaches displaying a GUI element in response to user input specifying a data source.

It would therefore have been obvious to one of ordinary skill in the art, having the teachings of Risberg, Kodosky, and Office 97 before him at the time the invention was made, to modify the application taught by Risberg and Kodosky, such that instead of creating a GUI element and then selecting a data source with which to associate the GUI element, the data source is first selected and then in response, a GUI element associated with the data source is displayed, as is done in Office 97. It would have been advantageous to one of ordinary skill to utilize such a combination because a more efficient means for configuring the active document to display data from a data source results; instead of having to create a GUI element and then select a data source with which to populate the GUI element, as is done by Risberg and Kodosky, with the combination of Risberg, Kodosky, and Office 97, a user simply has to select a data source – the GUI element is automatically created in response.

Regarding claim 19, the page fragment disclosed by Risberg, as modified by the teachings of Office 97, is displayed and configured in response to specifying a data source, whereby as shown above, the page fragment resultantly displays data received from this data source. As shown above, the user specifies a data source either by highlighting the particular data source, or by using a dialog box. It is interpreted that the other graphical elements disclosed by Risberg are configured by similar means. Thus these GUI elements are automatically configured without user programming and without the user input specifying source code.

Regarding claims 20 and 21, the combination of Risberg, Kodosky, and Office 97 teach a method whereby, as is shown above, an application is used to display and configure a GUI element to receive and display data from a specific data source. Such a GUI element is displayed and configured in response to specifying a data source. As shown above, the user specifies a

data source either by copying and pasting the particular data source, or by using a dialog box. In response, a GUI element is displayed which receives and displays data from this data source, as is shown above. As further taught by Office 97, this particular GUI element is based on the data source. For example, figure 3.9 on page 46 shows a GUI element, which is displayed in response to the specification of a spreadsheet data source. As shown in figure 3.9, this GUI element is a table. Figure 14.2 on page 382 on the other hand shows a GUI element, which is displayed in response to the selection of a document data source. As shown in figure 14.2, this GUI element simply comprises the text of the source document. Thus the GUI element is different based on the data source; a spreadsheet data source is displayed via a table-like GUI element, while a document data source is displayed via textual GUI element. Consequently, it is understood that the combination of Risberg and Office 97 described above teaches automatically determining an appropriate GUI element to display, based on the specified data source or target, wherein this determined GUI element is displayed automatically. Concerning claim 21, Office 97 further teaches that the particular GUI element is based on the data from the data source. For example, the GUI element in figure 3.9, which as described above is displayed in response to the selection of a spreadsheet data source, is a table comprising 1 column and 3 rows. Figure 14.8 on page 387 also shows a GUI element which is displayed in response to the selection of a document data source. As shown in figure 14.8, this GUI element is a table comprising 4 columns and 12 rows. Comparing figure 14.6 on page 386 and figure 3.7 on page 44, which respectively show the spreadsheet data sources for the above described GUI elements, it is noted that the source which is displayed in the larger GUI element, i.e. the GUI element of figure 14.8, is the spreadsheet comprising more data rows and columns (that of figure 14.6). Thus the GUI

element is different based on the data of the data source; a spreadsheet source with may columns and rows of data is displayed in a larger GUI element than a spreadsheet source with fewer columns and rows of data. Consequently, it is understood that the combination of Risberg, Kodosky, and Office 97 described above teaches receiving data from the data source, and automatically analyzing the received data to determine a GUI element operable to indicate the received data.

As for claim 22, the combination of Risberg, Kodosky, and Office 97 teach a method whereby, as is shown above, an application is used to display and configure a GUI element to receive and display data from a specific data source. Such a GUI element is displayed and configured in response to specifying a data source. The data received and displayed from the data source is in one of many different possible formats. For example, the data may be spreadsheet data or text data, among others, as is described above. In any case, since the data received from the data source may be in one of many different formats, which must be analyzed in order to ascertain how to display the data, it is understood that the data must be in a self-describing format. In other words, it is understood that the data itself at least partially describes how it is to be displayed. Consequently, the combination of Risberg, Kodosky, and Office 97 described above teaches that the data received is in a self-describing format, and wherein a GUI element is automatically determined that is operable to indicate this data.

With respect to claim 23, Risberg discloses that a data source may be specified by mouse actions equivalent to a drag-and-drop technique (see column 11, lines 46-64). Consequently, the combination of Risberg, Kodosky, and Office 97 described above teaches receiving user input

specifying a data source in response to a drag-and-drop user interface technique performed by the user.

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Blaine Basom whose telephone number is (703) 305-7694. The examiner can normally be reached on Monday through Friday, from 8:30 am to 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Cabeca can be reached on (703) 308-3116. The fax phone number for the organization where this application or proceeding is assigned is (703) 746-7238.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is 305-3900.

btb

JOHN CABECA
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER